

UNITED STATES PATENT APPLICATION

FOR

Method and Apparatus for Manipulating MPEG Video

INVENTOR: Inching Chen

Prepared by:

Blakely, Sokoloff, Taylor & Zafman
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, California 90025
(408) 720-8598

Attorney's Docket No. 042390.P9234

"Express Mail" mailing label number: _____

Date of Deposit: _____

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Assistant Commissioner for Patents, Washington, D. C. 20231

(Typed or printed name of person mailing paper or fee)

(Signature of person mailing paper or fee)

(Date signed)

042390.P9234

Method and Apparatus for Manipulating MPEG Video

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

This invention relates to the field of MPEG video, more specifically to a method and apparatus for manipulating and displaying MPEG video.

BACKGROUND INFORMATION

Figure 1 (Prior Art) is a diagram of the structure an MPEG video stream. An MPEG video is an ordered stream of bits with special bit patterns marking the beginning of a section. A section corresponds to MPEG's layered structure. The layers, or levels, are sequence 101, groups of pictures (GOPs) 102, individual pictures 103, slices 105, macroblocks (MBs) 107, and blocks 109 as shown in **Figure 1** (*Overview of MPEG*, Berkley Multimedia Research). Of course, the lowest layer is the pixel layer.

In MPEG video, the layers are structured in the following manner: a sequence 101 is composed of GOPs 102; a GOP 102 is composed of pictures 103; a picture 103 is divided into slices 105; a slice 105 is composed of MBs 107; and MBs 107 are composed of blocks 109 (*Overview of MPEG*, Berkley Multimedia Research). Each picture 103 is normally encoded into one of three types of pictures, an intra coded picture (I-picture) 110, a predictive coded picture (P-picture) 112, and a bi-directionally predictive coded picture (B-picture) 114. The I-picture 110 can be decoded independently because it uses only spatial correlation within the picture. P/B-pictures 112,114 take advantage of temporal correlation to achieve a higher degree of compression by only coding the

differential data. As a result, neither P nor B pictures 112,114 can be independently decoded (*Overview of MPEG*, Berkley Multimedia Research).

Similar to pictures, there are three types of macroblocks 105: I macroblocks (I-MB), P macroblocks (P-MB), and B macroblocks (B-MB). An I-picture 110 is coded using I-MBs only. While all three types of macroblocks 105 are used to code a B-picture 114, only I-MB and P-MB are used in coding a P-picture 112 (*Overview of MPEG*, Berkley Multimedia Research).

MPEG video has been widely used in various standards, such as DTV, DVD, and DVB. Although MPEG video decoding is currently done mostly in hardware for real-time performance, conventional hardware for MPEG video decoding is limited to the resolution of HDTV and is not capable of decoding ultra high-resolution MPEG video for a display wall.

For example, the University of Minnesota uses high-end equipment for their Power Wall. Behind the Power Wall, two SGI Power Onyx2's using multiple InfiniteReality graphics engines are controlled by a computer with a Silicon Graphics Indy control processor. The Power Wall projects a 3200X2400 resolution display 8 feet wide and 6 feet tall from 4 rear-projection monitors.

An attempt to use software for MPEG decoding is discussed in "*Real-Time Parallel MPEG-2 Decoding in Software*", Bilas, Fritts, and Singh, 11th Int'l Parallel Processing Symposium. Bilas et al discusses 2 software decoding approaches using a 16 processor Silicon Graphics Challenge multiprocessor (a fairly expensive multiprocessor) (*Real-Time Parallel MPEG-2 Decoding in Software*, Bilas, Fritts, and Singh, 11th Int'l Parallel Processing Symposium). First, Bilas et al uses a coarse approach by exploring

the picture level. Second, Bilas et al uses a fine-grained approach at the slice level. In the first approach, Bilas et al assigns a GOP to a processor as a task. In the second approach, Bilas et al assigns a slice to a processor as a task.

Another software approach implemented at Princeton University employs multiple interconnected personal computers (PCs) which use commodity graphics adapters for low cost. In this method, a mismatch exists between what a PC decodes and what the PC is responsible for displaying. To illustrate, PC A would decode slices representing regions to be displayed by PCs A and B. PC B must wait until node A has decoded the slices. Once PC A completes decoding, then PC A transmits the decoded data to PC B for B to display its data. The other problem is load balancing, i.e., when many slices require decoding, but only a few are to be decoded by a PC. The load imbalance occurs when the number of slices cannot be evenly distributed to all participating PCs. For example, there are 68 slices in a HDTV resolution, 1920x1088. Two of the six PCs in group A get to decode 12 slices, each, whereas the remaining four PCs in group B decode a total of only 11 slices. As a result, not all PCs in group B are busy.

Currently, very little work exists for displaying MPEG video at the high resolution of a display wall. Hardware implementations and some software implementations require costly, specialized hardware. The software implementations using conventional PCs include problems such as load balancing and communication overhead between PCs.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 (Prior Art) is a diagram of the structure an MPEG video stream.

Figure 2 is a diagram for partially decoding a picture to capture a region of

5 interest (ROI).

Figure 3 is a conceptual diagram of the "push method" of manipulating MPEG video with Regions of Interest (ROI's), according to one embodiment of the invention.

Figure 4 is a flow chart of the client's tasks in the "push method" of manipulating MPEG video, according to one embodiment of the invention.

10 **Figure 5** is a flow chart of the node PC thread of the "push method" at the picture level according to one embodiment of the invention.

Figure 6 is a flow chart of a node PC processing an I/P picture in the "push method" of manipulating MPEG video according to one embodiment of the invention.

15 **Figure 7** is a flow chart of a node PC processing a B picture in the "push method" of manipulating MPEG video according to one embodiment of the invention.

Figure 8 is a conceptual diagram of the "pull method" of manipulating MPEG video, according to one embodiment of the invention.

Figure 9 is a flow chart of the client's tasks for the "pull method" of manipulating MPEG video according to one embodiment of the invention.

20 **Figure 10** is a flow chart of the client thread for the "pull method" of manipulating MPEG video according to one embodiment of the invention.

Figure 11 is a flow chart for determining the conversion of a P/B-macroblock to an I-macroblock according to one embodiment of the invention.

DETAILED DESCRIPTION

In the following description, numerous specific details are set forth to provide a thorough understanding of the invention. However, it is understood that the invention may be practiced without these specific details. A method and apparatus for manipulating and displaying MPEG video is described.

Figure 2 is a conceptual diagram for partially decoding a picture to capture a region of interest (ROI). The ROI is an area of a picture, smaller than the entire picture, which is to be treated separately by a given device (e.g., to be displayed by the device, to be reencoded by the device, etc.). In **Figure 2A**, an ROI 201 of an MPEG encoded picture 202 is identified. In **Figure 2B**, a first slice 203 partially falling within the ROI 201 of the MPEG encoded picture 202 is decoded into multiple macroblocks 204. A set 212 of the multiple macroblocks 204 for the part of the slice falling within the ROI 201 are further decoded. In **Figure 2C**, a second slice 206 partially encompassed by the ROI is decoded into multiple macroblocks 207. In **Figure 2D**, a third slice 209 of the picture 202 is not decoded because it does not fall within the region of interest 201. In **Figure 2E**, a last slice 210 of the picture 202 also does not fall within the region of interest 201 and therefore is not further decoded. When combined, the set of macroblocks 212 and 215 in **Figure 2E** form the ROI 201. The ROI is located by partially decoding a picture without fully decoding the entire picture, thus reducing computational complexity and increasing work efficiency. In addition, a system may only extract information relevant to its ROI, and not other ROI's.

Figure 3 is a conceptual diagram of the "push method" of manipulating MPEG video with Regions of Interest (ROI's), according to one embodiment of the invention. In

the "push method", a client computer 301 broadcasts an MPEG stream 302 over a network to node computers 303-308 connected to the network. The nodes 303-308 decode their ROI's 309-314 from the MPEG stream 302. The node computers 303-308 then display their ROI's 309-314 in synchronization. Continuous synchronized display of the ROI's 309-314 collectively display the MPEG video stream 302.

Figure 4 is a flow chart of the client's tasks in the "push method" of manipulating MPEG video, according to one embodiment of the invention. At block 401, a client sends an attribute file to all nodes connected to the client over a network. After sending the attribute file, the client then begins to broadcast the MPEG stream to each of the node computers at block 402. At the same time, the client begins a synchronization task at block 403. The client's synchronization task at block 403 includes waiting for signals from the nodes on the network. A signal indicates that a node has decoded its ROI. When the client receives the signals from all desired nodes, the client sends a display command to the desired nodes. While the client broadcasts the MPEG stream over the network, it continuously performs the synchronization task. The dashed line 405 indicates the coordination between the synchronization task at block 403 and the computer sending MPEG streams at block 402. The client will not overload the node with MPEG data if the node does not have the capacity to handle any further MPEG data.

Figure 5 is a flow chart of the node PC thread of the "push method" at the picture level according to one embodiment of the invention. At block 501, a node receives an attribute file. The node then begins to receive MPEG encoded pictures at block 505 from the client. The node then determines if the picture is an I/P picture or a B-picture at block 506. At block 508, the node processes I/P-pictures. At block 507, the node processes B-

pictures. At block 520, the node determines if the buffer space is full. If the buffer space is full, then the node waits until space has been freed for the processed picture at block 525. If space is available or once space is made available, the node buffers a region of interest from the picture at block 509. At block 512, the node gets the next picture, while
5 at the same time waiting for a display command from the client at block 515. Once the node receives the display command from the client at block 517 the node displays the ROI.

Figure 6 is a flow chart of a node PC processing an I/P picture in the “push method” of manipulating MPEG video according to one embodiment of the invention.

10 The node gets the first slice of the picture at block 601. The server then determines if the slice is within the region of interest or a reference region in block 602. A reference region is an area surrounding an ROI. This area contains information relevant to the ROI. If at least part of the slice is not within the region of interest or its reference region,
15 then the node discards the slice at block 603, gets the next slice of the picture at block 604, and control returns to block 602. If at least part of the slice is determined to be within the ROI or the ROI's reference region, then the node gets the first macroblock from the slice at block 605. The node then determines at block 606 whether the macroblock is within the region of interest or the reference region. If the macroblock is not within the region of interest or the reference region, then at block 608 the node
20 determines if it has reached the end of the slice. If at block 608, it is determined that the node is not done with the slice, then at block 609 the node gets the next macroblock of the slice and control returns to 606. If at block 606 it is determined that the macroblock is within the region of interest or the reference region, then the node decodes the

macroblock at block 607 and control passes to block 608. If at block 608, it is determined that the node is done with the slice, then at block 610, the node determines if it has completed decoding its region of interest. If the node has not completed its region of interest, then it proceeds to get the next slice of the picture at block 604. If the node
5 has completed decoding its ROI, then at block 611, it sends a “done” signal to the client.

Figure 7 is a flow chart of a node PC processing a B picture in the “push method” of manipulating MPEG video according to one embodiment of the invention. The node gets the first slice of the picture at block 701. The server then determines if the slice is within the region of interest in block 702. If at least part of the slice is not within the
10 region of interest, then the node discards the slice at block 703, gets the next slice of the picture at block 704, and control returns to block 702. If at least part of the slice is determined to be within the ROI, then the node decodes the first macroblock from the slice at block 705. The node then determines at block 706 whether the macroblock is within the region of interest. If the macroblock is not within the region of interest, then at
15 block 608 the node determines if it has reached the end of the slice. If at block 708, it is determined that the node is not done with the slice, then at block 709 the node gets the next macroblock of the slice and control returns to 706. If at block 706 it is determined that the macroblock is within the region of interest, then the node decodes the macroblock at block 707 and control passes to block 708. If at block 708, it is
20 determined that the node is done with the slice, then at block 710, the node determines if it has completed decoding its region of interest. If the node has not completed its ROI, then it proceeds to get the next slice of the picture at block 704. If the node has completed decoding its ROI, then at block 711, it sends a “done” signal to the client.

The "push method" enables parallel distributed MPEG display in a multiple system environment with software decoding. Since each node PC is only responsible for a defined region of each MPEG picture of the MPEG stream, the node PCs do not need to communicate with each other. The reduction in network traffic is accompanied by an evenly distributed workload and higher utilization of the node PCs. Each node will fully decode only its ROI. Once a node PC completes decoding a ROI, the PC can begin decoding the ROI for one or more subsequent pictures, whether or not the display signal has been received (of course, based on buffer memory availability). Furthermore, the "push method" does not require high-end graphics equipment. The client and nodes can be conventional PCs.

According to another embodiment of the invention, a "pull method" of manipulating MPEG video is described. **Figure 8** is a conceptual diagram of the "pull method" of manipulating MPEG video, according to one embodiment of the invention. In the "pull method", a client computer 801 transmits multiple MPEG compliant substreams 802- 807 over a network. Each substream 802-807 corresponds to a spatial location across the sequence of pictures comprising the MPEG stream. The spatial location is less than an entire picture. Each of the MPEG compliant encoded substreams 802 through 807 are transmitted to a specific node 808-813 connected to the network. Each node 808-813 decodes its substream 802-807 and displays the decoded substream 814-819. The client computer 801 can synchronize display of the decoded substreams 814-819 to display the MPEG stream 820. The MPEG compliant substreams 802-807 are created by the client computer at 801 from the MPEG stream 820. The client 801 may only create a single substream if only one ROI is desired.

This manipulation of an MPEG video stream to create a completely independent substream eliminates the problem of communication overhead between the nodes. In addition, an MPEG compliant substream corresponding to an ROI can provide a variety of new applications for MPEG video. A user may only be interested in one region of an MPEG video. A system could create a single substream focusing on that region from the MPEG video. This single substream can be scaled or manipulated further depending on the user's requirements. Some application may require manipulation of independent substreams from different MPEG streams. If desired, a single substream could be made which tracks a roaming ROI.

Figure 9 is a flow chart of the client's tasks for the "pull method" of manipulating MPEG video according to one embodiment of the invention. At block 901, the client begins to make MPEG substreams from a main MPEG stream. While still making MPEG substreams in block 902, the client begins to send the substreams over a network at block 903. While sending the substreams and making additional substreams in with a lock-step approach, the client begins performing a synchronizing task at block 904.

Figure 10 is a flow chart of the client thread for the "pull method" of manipulating MPEG video according to one embodiment of the invention. At block 1001, the client gets an MPEG encoded picture from an MPEG stream. If at block 1002, the MPEG encoded picture is an I-picture, then the client decodes the I-picture at block 1003 to the macroblock level. At block 1004 the client separates the macroblocks into data structures according to one or more regions of interest. The client then uses the macroblocks in a data structure corresponding to one or more ROI's to construct a corresponding MPEG compliant I-picture at block 1005. According to one embodiment

of the invention, new pictures are constructed slice by slice. For I-pictures, a new slice is constructed by putting together all the I-macroblocks within the ROI. The Q-value at the slice level needs to be updated by the Q-value used to code the first macroblock. In addition, the VLC code of the DC components for Y, Cr, and Cb of the first macroblock, require updates because they are coded losslessly by a DPCM technique. At each start of a slice, DC_DCT_PRED is reset and a new DC_DIFF value is calculated.

It is determined at block 1006 if the MPEG encoded picture is a P/B-picture. If the picture is a P/B-picture, then the client decodes the picture to the macroblock level at block 1007. It is determined at block 1009 if the MB is a P/B-MB. For those MBs that are P/B-MB, the client at block 1010 determines if each of the P/B-macroblocks require conversion to an I-macroblock. One exemplary embodiment of block 1010 is later described in more detail with reference to **Figure 11**. At block 1012, the client separates macroblocks into data structures according to one or more ROI's. At block 1015, the client then uses the MBs of each of the region(s) to construct an MPEG compliant P/B-picture. According to one embodiment of the invention, new pictures are constructed slice by slice. Like an I-picture, the Q-value of the first macroblock needs to be updated at a slice or macroblock level. If the first macroblock of the slice is an old I-macroblock, then the DC_DIFF values need an update just like an I-picture. However, if the first macroblock is a new I-macroblock and is followed by an I-macroblock, the DC_DIFF values of the following I-macroblock need to be updated. Similar to the DC_DCT_PRED, the MV is coded differentially with respect to previous decoded MV, and uses VLC. However, there are some differences. Forward MV and backward MV should be treated independently, because a B-MB can be decoded with forward MV

and/or backward MV. A skipped MB has different effects on P- and B-pictures. In a P-picture, the PRED_MV is reset to zero when a MB is skipped. However, the PRED_MV are not affected in a B-picture.

Figure 11 is a flow chart for determining whether the conversion of a P/B-

macroblock to an I-macroblock is required (block 1012), according to one embodiment of the invention. At block 1101 the client determines if a P/B-macroblock's reference macroblock is out of the region of interest. A reference MB contains information needed by a MB, such as motion compensation information. If the reference macroblock is out of the region of interest, then the client converts the P/B-MB to an I-MB at block 1105.

In one embodiment of the invention, a MB is converted to an I-MB by decoding the P/B-MB to the pixel level and then encoding the pixels as MPEG compliant I-MBs. All MBs converted to I-MBs are encoded as if the first MB of a slice. If the P/B-MBs reference MB is not beyond the ROI, then the client determines if the P/B-macroblock is a skipped macroblock at block 1102. If the P/B-MB is not a skipped MB, then at block 1106 the client stores the P/B-MB into a data structure according to region of interest. If the P/B-MB is a skipped MB, then it is determined if the P/B-MB will be at the beginning or end of a new slice at block 1103. If the P/B-macroblock will be at the beginning or the end of a new slice, then the P/B-macroblock is converted to an I-MB at block 1105. If the P/B-macroblock will not occur at the beginning or end of a new slice, then it is determined if the skipped P/B-macroblock will follow a new I-macroblock in a B-picture in block 1104. If the skipped macroblock does not follow a new I-macroblock in a B-picture, then the client stores the P/B-macroblock in a data structure according to region of interest at block 1106. If the skipped P/B-macroblock does follow a new I- macroblock in a B-

picture, then the client converts the skipped P/B-macroblock to an I-macroblock at block 1105. Converted I-macroblocks are stored after conversion into one or more data structures according to region of interest at block 1106.

Like the "push method", the "pull method" also manipulates MPEG video enabling parallel distributed MPEG video display in a multiple system environment with software decoding. As in the "push method", the "pull method" doesn't require communication between the nodes. In addition, network traffic is further reduced because the client transmits substreams for one or more ROI's instead of the entire MPEG stream to each node. The nodes can use a conventional MPEG decoder since the substreams are MPEG compliant. Moreover, the workload can be evenly balanced across the nodes because the nodes are only responsible for a substream representing their ROI.

The techniques shown in the figures can be implemented using code and data stored and executed on computers. Such computers store and communicate (internally and with other computers over a network) code and data using machine-readable media, such as magnetic disks; optical disks; random access memory; read only memory; flash memory devices; electrical, optical, acoustical or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.); etc. Of course, one or more parts of the invention may be implemented using any combination of software, firmware, and/or hardware.

While the invention has been described in terms of several embodiments, those skilled in the art will recognize that the invention is not limited to the embodiments described. The MBs in the "pull method" can be stored in a variety of data structures, e.g., trees, tables, lists, hybrid trees, etc. The system may be set up to store MBs for

